TIME-SHARING ON COMPUTERS

This technique, whereby a computer serves a large number of people at once, does more than save time and money. It sets up a dialogue between user and machine and allows communication among users

by R. M. Fano and F. J. Corbató

The history of the modern computer has been characterized by a series of quantum leaps in our view of the machine's possibilities. To mention only two of the crucial advances, the application of electronics, vastly increasing the computer's speed of operation, and later the invention of special languages, facilitating communication with the machine, each in its turn opened new vistas on the computer's potentialities. Within the past few years the technique called time-sharing has again stimulated the imagination. It has created an unexpected new order of uses for the computer.

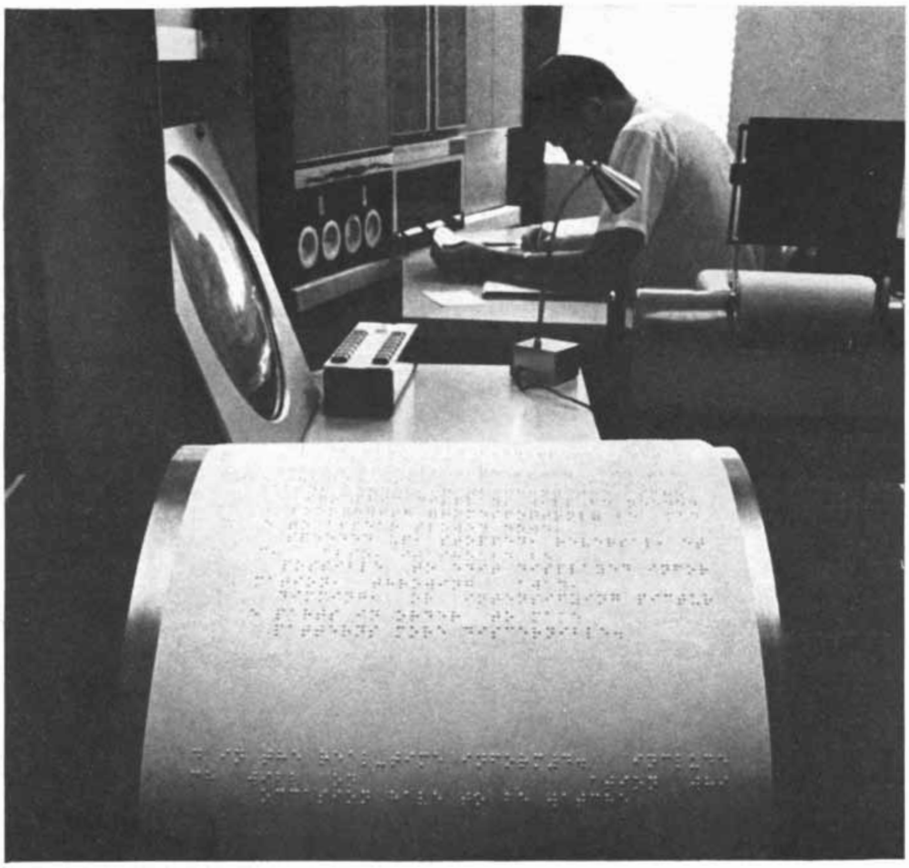
At first thought time-sharing seems simply a convenience: a means of allowing fuller use of the machine by more people and of saving time for the users. In practice, however, experiments with the technique have demonstrated a wide range of more interesting possibilities. It enables the user to conduct a continuous dialogue with the machine and in effect makes the computer his intellectual assistant. Further, the system makes it possible for the users to carry on a discourse with one another through the machine, drawing on its large store of knowledge and its computing speed as they do so. The time-sharing computer system can unite a group of investigators in a cooperative search for the solution to a common problem, or it can serve as a community pool of knowledge and skill on which anyone can draw according to his needs. Projecting the concept on a large scale, one can conceive of such a facility as an extraordinarily powerful library serving an entire community—in short, an intellectual public utility.

It was Christopher Strachey, the author of the article on system analysis and programming in this issue, who first proposed (in 1959) a time-sharing system. The large, expensive computing machines had become far removed from their users, both in time and in distance. An applicant in effect had to deliver his problem or program to a receptionist and then wait hours or sometimes days for an answer that might take the machine only seconds or even less time to produce. The computer, working on one program at a time, kept a queue of users waiting for their turn. If, as commonly happens, a submitted program contained a minor error that invalidated the results, the user often had to wait several hours for resubmission of his corrected program. Strachey suggested that the rapidity of a computer's operations made all this waiting unnecessary. By segregating the central processing operations from the time-consuming interactions with the human programmers, the computer could in effect work on a number of programs simultaneously. Giving only a few seconds or often less than a second at a time to each program or task, the machine could deal with many users at once, as if each had the machine to himself. The execution of various programs would be interspersed without their interfering with one another and without detectable delays in the responses to the individual users.

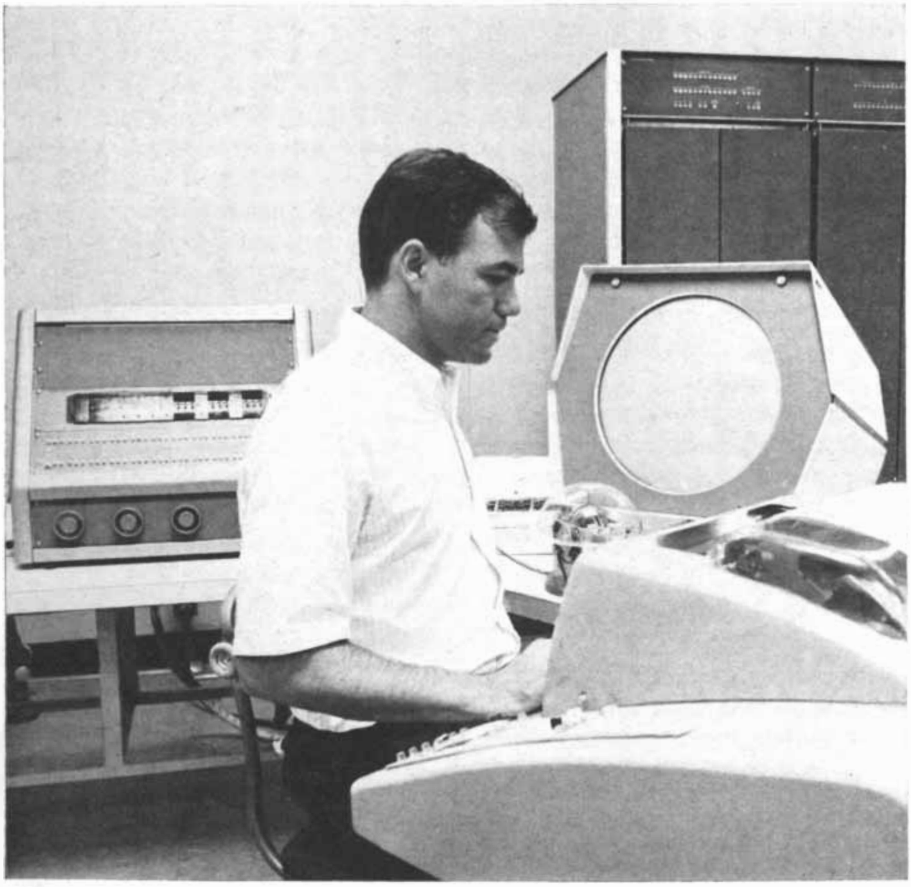
The Computation Center at the Massachusetts Institute of Technology quickly took up Strachey's suggestion. By November, 1961, the center had implemented and demonstrated a first model of the Compatible Time-Sharing System, using an International Business Machines Corporation 709 computer. Two years later an improved version of this system was operating on two IBM 7094 computers, one at the Computation Center and another at M.I.T.'s Project MAC (an acronym that has been variously translated as standing for multiple-access computer, machine-aided cognition or man and computer). By that time three other time-sharing systems had been developed: at Bolt, Beranek and Newman, Inc., at M.I.T.'s Research Laboratory of Electronics and at the System Development Corporation, and several more have since been developed at other research institutions.

Inherent in the time-sharing concept is a system of multiple direct connections to the computer from many points, near and far. At M.I.T. there are now 160 such stations, each with a teletypewriter that enables the user to enter his message directly in the computer's input and to receive its replies. These stations are installed in various offices and laboratories on the M.I.T. campus and in

PROJECT MAC time-sharing system at the Massachusetts Institute of Technology has 160 terminals on the M.I.T. campus and nearby and is also available from distant terminals. As many as 30 terminals can be connected at one time, with each user carrying on a direct and in effect uninterrupted dialogue with the computer. The terminals, 30 of which are shown on the opposite page, are for the most part simple teletypewriters such as the IBM 1050 (*6*) and Teletype models 33 (*19*), 35 (*5*) or 37 (*10*). Some are in offices, some in large "pool" rooms, some in laboratories and a few in private homes (*1*). In addition to students and staff members doing their own research, the users shown here include secretaries preparing papers for publication (*13*), authors Fano (*8*) and Corbató (*24*) and a psychiatrist at the Massachusetts General Hospital (*18*). More elaborate terminals are shown on the next page.



SPECIALIZED TERMINAL of Project MAC has a small computer, the Digital Equipment Corporation's PDP-8, operating a braille printer (*foreground*) for a blind staff member.



ANOTHER TERMINAL is the "Kluge" display system developed by M.I.T.'s Electronic Systems Laboratory. It has a control unit, display screen, light pen and other equipment.

the homes of some of the research staff and faculty members. Through a private branch exchange each station can by dialing reach either the Project MAC computer or the one in the Computation Center. Moreover, the Project MAC installation is connected to the teletype networks of the Bell System and Western Union, so that access to the computer can be had from thousands of terminals in the U.S. and abroad. Thus the two computer systems at M.I.T. are being used daily by a large and varied community, with each of them providing prompt response for up to 30 simultaneous users. The systems constitute an operating model of the information utility that John McCarthy, the author of the introduction to this issue, described in 1961 in an address picturing computer services of the future.

For professional programmers the time-sharing system has come to mean a great deal more than mere ease of access to the computer. Provided with the opportunity to run a program in continuous dialogue with the machine, editing, "debugging" and modifying the program as they proceed, they have gained immeasurably in the ability to experiment. They can readily investigate new programming techniques and new approaches to problems. The bolder exercise of the imagination encouraged by the new system has resulted not only in more flexibility in attacks on problems but also in the undertaking of important new researches in a variety of areas.

Let us now examine the operation of a time-sharing system. Taking the M.I.T. Compatible Time-Sharing System (CTSS) as our model, we shall first present a sample of what it can do, using a program dialogue for illustration, and then describe the anatomy and machinery of the system in schematic terms.

To begin with, the system contains a large store of information—supervisory and utility programs, language-translating facilities, a library of subroutines and so on—adding up to nearly a million computer words, which is equivalent to about 2,000 book pages crowded with nonredundant symbols. The basic content of the system is a set of some 100 programs, each of which is called into play by a specific command issued through a teletypewriter. They have to do with the ordinary operations of the system and involve communication, control of its various processes, the use and translation of computer languages and so forth. In addition to these 100 basic

programs the system contains a great variety of special programs that are also available for general use. To all this "public" information there is added a large amount of material consisting of individual users' private files of programs and information.

Consider, then, an illustrative dialogue between a user and the computer [*see illustrations on pages 132 through 134*]. The user introduces himself by giving the command "login" and stating the project he wants to work on and his name. The machine responds by printing the time of day (to the hour, minute and tenth of a minute), and the user is now called on to give his password. This has been found to be a highly important requirement: it is necessary to guard the privacy of each personal set of files and protect the information and programs from accidental or malicious alteration by someone else. (Experience has shown that some people are unable to resist the temptation to commit mischievous vandalism of that kind.) The printer is disconnected while the password is being typed so that no record of it appears on the print-out.

If the given password does not check with the person's name and problem number, or if he has exhausted his monthly allowance of time on the computer, or if the machine is already being used to capacity (the maximum number of people who can use the computer at one time in our present system is 30), the machine prints a message stating that access is not available. If access can be granted, the user is allowed to proceed with further commands. Before beginning his work he may ask for an accounting of the amount of time and storage space he has used up from his allotted quotas (as one illustration shows). After this housekeeping query the user goes to work on his problem. Here, in our simple illustration, he

COMPUTER ROOM at M.I.T. houses the IBM 7094 central processing unit, memories and other central units of the Project MAC system. Disk and drum files are visible in left background, central processor in right background, tape-storage units at right.

```
login t193 fano
W 2237.7
Password
 T0193  2859 LOGGED IN  05/27/66     2237.8 FROM 20000N

 CTSS BEING USED IS     MAC5A4
 LAST LOGOUT WAS  05/27/66    2237.0
R 2.833+.900



ttpeek
W 2238.2

 5/27  2238.2 TUSED =   .1

SHIFT       MINUTES
      ALLOTTED   USED
 1       60       3.8
 2       20       1.0
 3        5       0.
 4       30        .3

      STORAGE
DEVICE   QUOTA   USED
DISK       100      4

R 1.950+.350



ed mad
W 2238.8
INPUT:
start read and print data
      an"mean = (x+y)/2.
      gmean = (x?   gmean = sqrt.(x*y)
      print results g"amean, gmen
      transfer to start
      end of program

EDIT:
top
print 10

START      READ AND PRINT DATA
           AMEAN = (X+Y)/2.
           GMEAN = SQRT.(X*Y)
           PRINT RESULTS AMEAN, GMEN
           TRANSFER TO START
           END OF PROGRAM
END OF FILE REACHED BY:
PRINT 10
file demo
R 3.983+2.550



mad demo
W 2242.9
 THE FOLLOWING NAMES HAVE OCCURRED ONLY ONCE IN THIS PROGRAM.
 COMPILATION WILL CONTINUE.
      GMEAN
      GMEN
LENGTH 00066.  TV SIZE 00005.  ENTRY 00017
R 1.750+.650

ed demo mad
W 2243.3
EDIT:
locate gmen
change /gmen/gmean/
print
           PRINT RESULTS AMEAN, GMEAN
top
change /amean/armean/ 10
END OF FILE REACHED BY:
CHANGE /AMEAN/ARMEAN/ 10
file
R 3.800+1.300

mad demo
W 2244.7
LENGTH 00063   TV SIZE 00005.  ENTRY 00016
R 1.466+.416

loadgo demo
W 2245.0
EXECUTION.
x = 123.456, y = 234.567 *
X = 123.456, Y = 234.567 *

          ARMEAN =    179.011499,          GMEAN =    170.172569
 QUIT,
R 5.483+1.283

save demo
W 2245.9
R .866+.283
```

RUNNING DIALOGUE between man and machine is demonstrated by a computer print-out. The user (*lowercase letters*) announces himself; the computer (*uppercase*) gives the time. The user gives his password (which, to preserve privacy, is not printed) and the machine logs him in and reports the number of seconds used by the central processor in the exchange. In response to the command "ttpeek" the machine summarizes Fano's time and memory-space account. With "ed mad" Fano writes and edits a program in MAD language for computing the arithmetic and geometric means of any two numbers. (The symbols " and ? erase the preceding character or all the preceding characters in the line respectively.) The program is filed under the name "demo." The machine queries a typing error ("GMEN" for "GMEAN"), which is corrected, and "AMEAN" is changed to "ARMEAN." The program is translated and executed. After a test computation the execution is interrupted. The computer acknowledges the interruption ("QUIT") and is instructed to "save" the program.

writes, translates and executes a program to compute the arithmetic mean and the geometric mean of two given numbers. A command signified by "ed" (for "editing") brings into play a subsystem in the computer that accepts various editing instructions, so that the user can call on it to write his program in a particular language (here it is the language called Michigan Algorithmic Decoder, or MAD) and make alterations, corrections or other manipulations of the program text as directed. Typing errors have been introduced deliberately in our example to illustrate some of these editing functions. The corrections are carried out through a dialogue between the user and the computer. Finally the program is translated and tested on a pair of numbers. The user presses an "interrupt" button at that point to end the computation and the computer prints out "QUIT" in acknowledgment. The user then gives the command "save demo," which instructs the machine to file the program in what becomes a new private file named "demo saved." This file is stored permanently in the system's mass memory.

The owner can command the system to print out the list of files in his file directory by giving the instruction "listf." He may also give a command authorizing the system to allow other named users access to one of his files, and conversely may gain access to other private or public files he is permitted to use. Although a person given access to someone else's file is not usually allowed to change that file, he can copy its contents, file the information separately under his own name and then modify the data or program for his own use. This technique is employed by a second person to use the program for computing the arithmetic and geometric means of a pair of numbers [*see bottom illustration on opposite page*]. Another convenient feature of the time-sharing system that is illustrated allows the depositing of messages from one user to another in the computer. On logging into the computer a user may be informed by the machine that there is a message in his "mail box," and the computer will then print the message on command.

To illustrate the editing capabilities of the system we have added a sample print-out of a paper delivered at a conference, together with the commands that enabled the computer to present it in the desired typographical form. The system includes a special facility for

editing English text, and this facility has been used in the preparation of technical reports and other publications.

Not the least useful feature of the Compatible Time-Sharing System is the fact that it carries its own set of instructions to its users. Stored in its mass memory is the manual describing the system; this is indexed by a table of contents listing the various services and sections in the reverse chronological order of their addition to the system; that is, the latest are listed first. Thus a user can readily check at any time to see whether or not his copy of the manual is up to date and can then obtain a print-out of any new or modified sections.

To explain the workings of the system we have focused on the dialogue carried on between the user and the computer through the medium of printed commands and responses. The Project MAC system also includes two display stations with facilities for light-pen drawing on a cathode ray tube and for viewing the projection of continuously rotating three-dimensional objects. This equipment has been used by Cyrus Levinthal for studying the structure of biological molecules [see "Molecular Model-building by Computer," by Cyrus Levinthal; SCIENTIFIC AMERICAN, June.]

Of the anatomy and internal operations of the M.I.T. time-sharing system we can only give a schematic outline. It employs a very large and complex installation, built around an IBM 7094 computer and containing in addition a number of special units [*see illustration on page 135*].

The heart of the system is a complex of programs called "the supervisor." It coordinates the operation of the various units, allocates the time and services of the computer to users and controls their access to the system. The allocation function includes scheduling of users' requests, transferring control of the central processor from one user to another, moving programs in and out of the core memory and managing the users' private files. Obviously the time allowance for each program-run must be closely regulated. If a program runs too long without interruption, other users will be kept waiting unduly; on the other hand, if the execution of a program is interrupted many times, the repeated movement of the program in and out of the core memory will entail a waste of time. We have adopted a time-allowance scheme based on task priorities that in

```
listf
W 2246.0

     6 FILES    24 RECORDS
 NAME1  NAME2 MOD NOREC   USED
  DEMO  SAVED 000    18 05/27/66
(MOVIE TABLE) 001     1
  DEMO    BSS 000     1
  DEMO    MAD 000     1
PERMIT   FILE 120     1
  FJCC (MEMO) 000     3 05/19/66

    28 LINKS
 NAME1  NAME2 MOD PROBN. PROGN. LNAME1 LNAME2
 BASIS  SAVED 104  T0173     44
CIRKIT  SAVED 000  T0113 CMFL04
CONVOL  SAVED 104  T0173     44
 CONVT    BSS 104
  DATA  SAVED 104
DATTOC (MEMO) 104  T0254   3212
DOCTOR  SAVED 144  T0109   2531
  GETF  SAVED 104  T0173     44
 GRPS1   DGET 144  T0263   32
- QUIT,
R 1.666+.300




permit demo saved 4 t100 385
W 2248.4
R .816+.266

edl permit mills
W 2248.9
 FILE PERMIT MILLS NOT FOUND.
Input
t 19?t193 2859 per "mits file demo saved to t100 385 mills

Edit
top

print 2

T193 2859 PERMITS FILE DEMO SAVED TO T100 385 MILLS
file
R 4.400+1.500

mail permit mills t100 385
W 2254.7
R 1.366+.433
```

DIRECTORY of Fano's personal files is printed out in response to the command "listf." It includes three files associated with the new "DEMO" program as well as "LINKS" to public and private files that Fano may use. Fano goes on to give permission for another user named Mills to use the new means-computing program and sends a message to Mills telling him so.

```
 login t100 mills
W 2255.4
Password
 YOU HAVE      MAIL    BOX
 T0100   385 LOGGED IN  05/27/66     2255.7 FROM 20000A

 CTSS BEING USED IS      MAC5A4
 LAST LOGOUT WAS  05/27/66    1555.3
R 2.766+.716

print mail box
W 2256.2

  MAIL    BOX      05/27  2256.3

FROM    T0193   2859 05/27 2254.7
T193 2859 PERMITS FILE DEMO SAVED TO T100 385 MILLS
R .683+.516

link demo saved t193 2859
W 2257.9
R 1.266+.433

resume demo
W 2258.0
x = 345.678, y = 456.789 *
X = 345.678, Y = 456.789 *


          ARMEAN =    401.233498,          GMEAN =    397.368725
 QUIT,
R .133+1.450

logout
W 2259.2
 T0100   385 LOGGED OUT 05/27/66    2259.3 FROM 20000A
 TOTAL TIME USED=      .1 MIN.
```

USER MILLS, logging in, is told that there is a message in his "MAIL BOX." After reading the message he asks that a link be established to the "demo saved" program that has been permitted to him. When this has been done, he asks the computer to "resume" the program, applying it to two new numbers he provides. The machine does so and prints the answers.

```
resume who
W 2300.6

MAC5A4 STARTED AT 1451.1 05/27

  BACKGROUND USED  142.6.      PERCENTAGE =
 17  USERS AT  2300.9 05/27

LINE   USER      NAME  GRP UNIT   TUSED TIMEON
 1 C0056 99995 FIBMON  0 (FIB)       .1 2217.9
 2 C0056 99999 DAEMON  0 DAEMON  37.6 1451.2
 4 T0269  8048 ENNING  3 200007   2.0 2115.9
 5 T0143   799    LIU  1 20000+   7.8 1923.7
 6. T0281  3712 MAURER  1 20000+   8.8 1805.0
 7 T0113  4619  WYLIE -1 600040    .8 2250.0
 8 T0193  2859   FANO  2 20000N    .7 2237.8
 9 T0269  8031 SSANNA  3 20000.   3.6 2044.0
10 T0234  1122 GARMAN  1 700168   1.4 2211.7
11 T0186  4288 INTOSH 15 20000+   5.9 2128.2
12 T0109  2531 ENBAUM  1 20000W   4.5 2125.9
13 T0145  3667 HITMAN  1 600038   1.8 2218.7
14 T0312  3047 NICHEL  1 20000Y   1.9 2222.5
15 T0335  4655 ULIANO -1 20000+    .1 2258.2
16 T0113  3556 EPHUIS  1 200000   3.3 2139.8
17 T0186  3187 MORRIS -1 100035    .4 2223.9
19 T0234  3308 WIDRIG  1 100001   1.4 2227.8

R 1.850+2.133
```

THROUGH A LINK to the system's public file, Fano asks for and receives a print-out of the system's current users, the time they logged in and the amount of time they have used.

```
typset fjcc
W 2303.3
Edit
top
print  20

.page
.header SOCIAL IMPLICATIONS OF ACCESSIBLE COMPUTING
.center
SOME THOUGHTS ABOUT THE SOCIAL
.center
IMPLICATIONS OF ACCESSIBLE COMPUTING
.space
.center
by
.space
.center
E. E. David, Jr.
.center
Bell Telephone Laboratories
.space
.center
R. M. Fano
.center
Massachusetts Institute of Technology
file
R 5.733+.916



runoff fjcc
W 2305.8
Load paper, hit return



                 SOME THOUGHTS ABOUT THE SOCIAL
              IMPLICATIONS OF ACCESSIBLE COMPUTING

                               by

                        E. E. David, Jr.
                  Bell Telephone Laboratories

                          R. M. Fano
             Massachusetts Institute of Technology


                            ABSTRACT

The pattern of our  business  and  private  lives  has  been
shaped by many important technological developments such  as
automobiles, electric power and telephones.   The  influence
of these products of technology was felt  when  they  became
available to a large segment of the population.  We are  now
at that stage with computers.

As  with  previous  products  of  technology, accessible
computing will undoubtedly benefit  society  but  will  also
face us  with  new  problems  and  new  frustrations.   The
underlying issues are very complex and they  deserve  prompt
and thoughtful consideration on the part of all of us.



logout
W 2307.9
 T0193  2859 LOGGED OUT 05/27/66    2308.0 FROM 20000N
 TOTAL TIME USED=      .9 MIN.
```

EDITING CAPABILITY of the system is illustrated by the machine's reproduction of the beginning of an article. The command "typset" calls up the program for editing and printing the text. Commands prefaced by a period, such as "center" and "space," are instructions on format. The command "runoff" produces a print-out in the specified format. Logging out, Fano learns that demonstration, which lasted 30.3 minutes, used .9 minute of computer time.

turn are determined initially by the amount of information that must be transferred into the core memory. The smaller the amount of information, the higher the initial priority the task is given. The time allowance is at least two seconds and doubles with each level of decreasing priority. If a task is not completed in its allotted time—or if a higher-priority task is waiting—it is interrupted and enough of the program moved out to a storage drum to make room in the core memory for the next task awaiting the processor's services. If the allotted time has been exhausted, the task's priority is lowered and a correspondingly doubled allocation of time is made. The interrupted task is then continued when its turn comes up again.

The user need not remain in communication with the system while his program is being run. He may write a program in collaboration with the machine, test it and, after he is satisfied that it is correct, instruct the internal supervisor to run the program for him and store the results in a file from which he can retrieve them later at his convenience. This arrangement, called FIB (for "Foreground-initiated Background"), is designed particularly for programs involving lengthy computations that do not require human intervention. The present system also allows, in effect, for the concurrent running of programs on a "batch" basis (that is, not time-sharing), but this facility is now largely superseded by FIB.

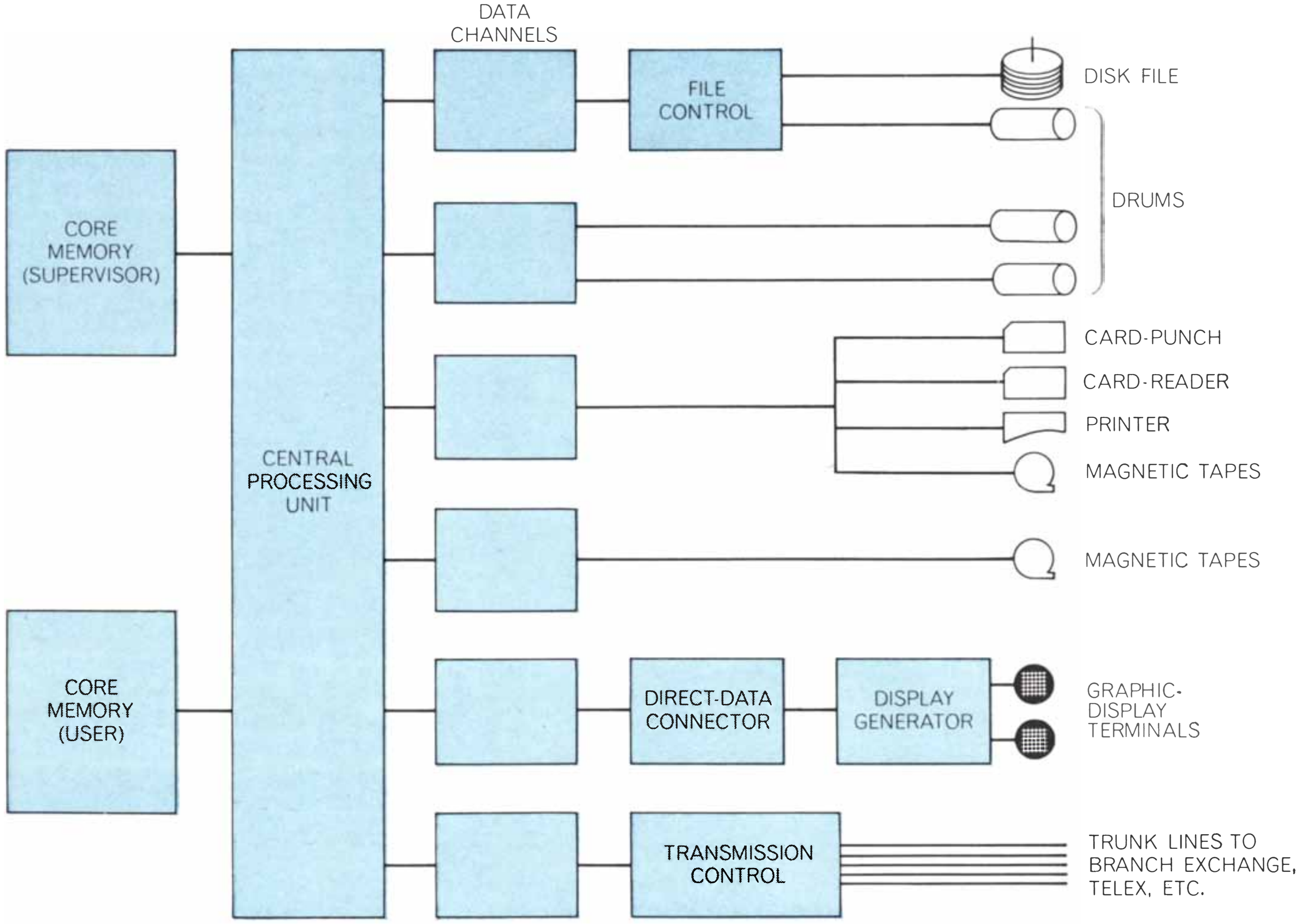
This, then, in sketchy outline, is the compatible time-sharing system we have been working with so far at M.I.T. It is only a precursor, of course, of systems that will be developed in the future. What improvements or advances are needed to create an installation that will serve a large community as a general public utility?

One obvious necessity is that the system provide continuous and reliable service. A public utility must be available to the community 24 hours a day and seven days a week without interruption. It should not shut down for accidents, repairs, maintenance, modifications or additions to the system. This implies, among other things, that the system should not depend completely on any one unit. It suggests that every part of the system should consist of a pool of functionally identical units (memories, processors and so on) that can operate independently and can be used interchangeably or simultaneously at all times [*see upper illustration on page*

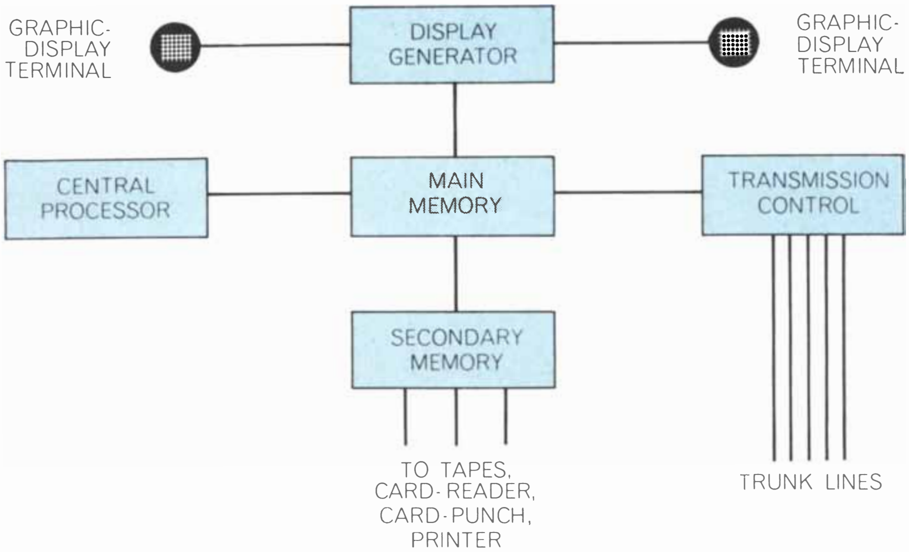
138]. In such a system any unit could be taken out of service for repair or maintenance during a period when the system load was low, and the supervisor would distribute the load among the remaining units. It would also be a simple matter to add units, without interrupting service, as the use of the system grew. Moreover, the availability of duplicate units would simplify the problem of queuing and the allocation of time and space to users.

A second need is more efficient use of the computer's time. In the Compatible Time-Sharing System, as in most conventional batch-processing systems, the central processor is idle for about 40 percent of the time because it must wait while programs and data are being transferred in and out of the core memory and while necessary information is being fetched from or written into the users' files. One way to reduce the processor's idle time would be to have at all times in the core memory several executable programs (instead of only one), so that as soon as the processor finishes a task or transmission of more data is required, it would find another task available. The computer art now presents a technique for producing this desirable situation without having to waste too much core memory to store entire programs waiting to be executed. A program can be divided into pages, each containing, say, only 1,024 words, and the core memory can be divided into logical blocks of the same size. Pages are transferred into core memory only when needed, if at all, so that tasks can be initiated with minimal use of precious memory space.

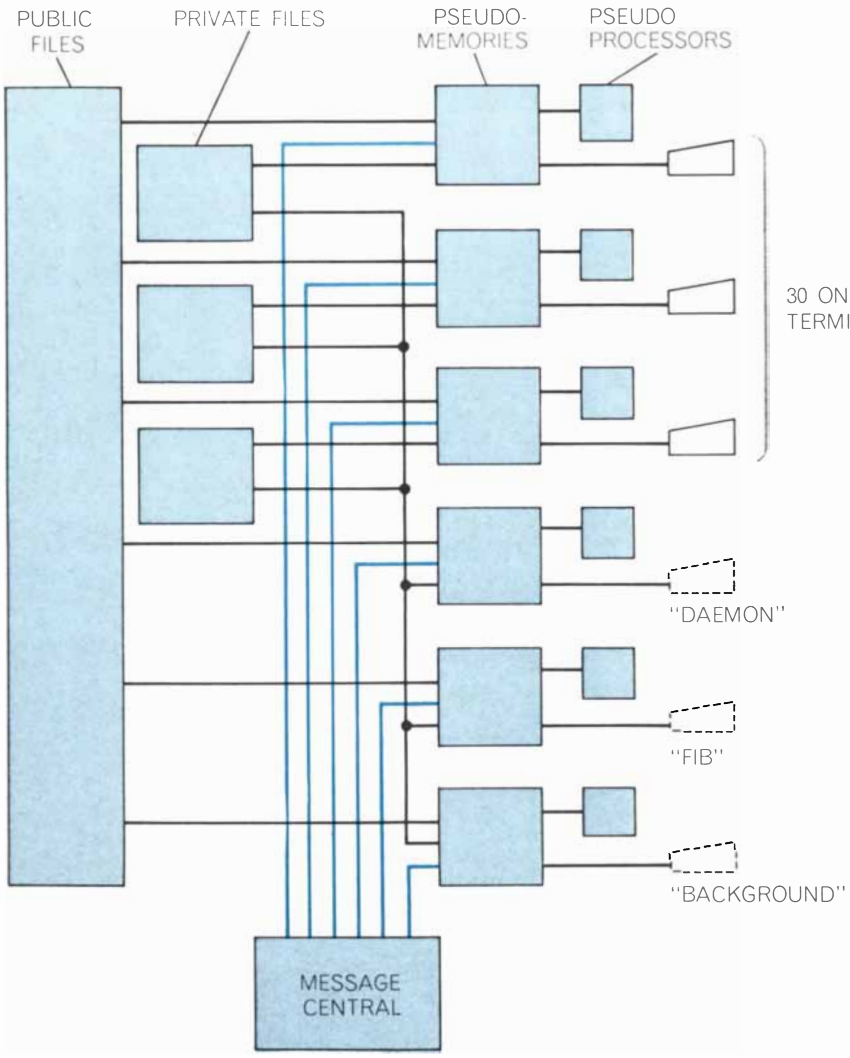
Another new technique, called program segmentation, has been advocated by Jack B. Dennis of M.I.T. to increase the ease and flexibility with which subprograms may be linked to form large programs. The process followed by a computer in executing a large program is similar to that followed by the reader of an article that refers to a section of another article that in turn refers to a chapter of a book, and so on. The traditional technique for linking subprograms is equivalent to having a clerk in the library make copies of all articles and books to be read, assemble them into a single volume, and translate all references into references to specific page numbers of the volume. This technique has the disadvantage that popular subprograms have to be copied and stored many times as parts of different programs. Moreover, programs, unlike articles and books, are often changed and new subprograms have to be incorporated. This is particularly true in a time-sharing system. With the technique of program segmentation the segments, or subprograms, retain their individual identity at all times. They are retrieved from mass storage only when the computer finds a reference to them during the execution of some other segment. Speed of retrieval, and particularly speed of access to individual words of a segment after the segment has been retrieved, is essential. For this purpose



PRINCIPAL ELEMENTS of the M.I.T. time-sharing computer installation are shown in a simplified schematic diagram. One of the two core memories is occupied by the supervisor program, which runs the system; the other is available to users. Files are moved into the core memory as needed from the disk and drum memories. The transmission control is actually a special-purpose computer.



SUPERVISOR has the effect of reducing the equipment layout diagrammed on the preceding page to the functional arrangement illustrated here. The main (core) memory, rather than the central processing unit, is in effect the central unit with which other units communicate; the various mass storage devices are in effect a single secondary memory.



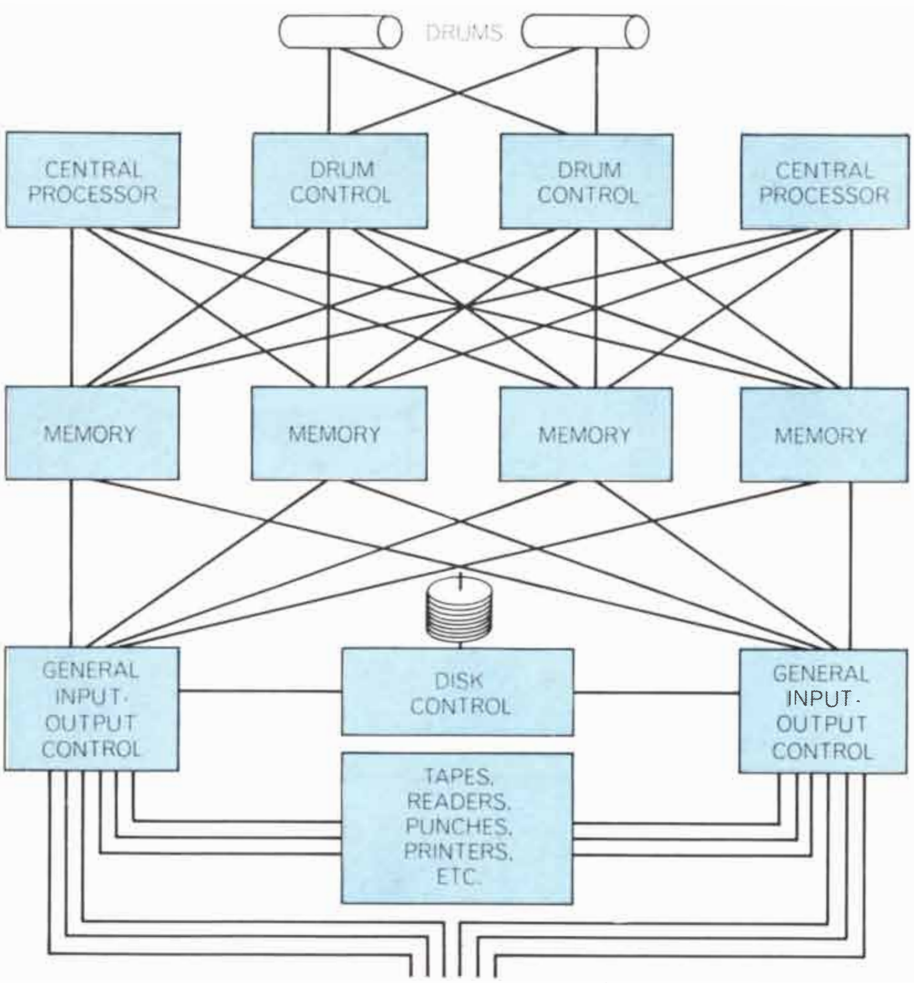
"USER'S VIEW" of the system is quite different. Each of the 30 on-line users has available, for all practical purposes, his own processor and memory. Each memory has in effect a capacity of 32,768 words and has access to public files as well as the user's own files. Messages can be exchanged through the message central. Three special pseudo-processors are available to the supervisor. "Daemon" copies files on tape. "FIB" takes over and executes programs for users who do not need to wait for lengthy answers. "Background" operates as a conventional computer, batch-processing large tasks fed into the central computer.

the computer must include special equipment features, and appropriate directories of all segments must be maintained. The position of the computer is then analogous to that of a user of an ideal automatic library who finds in his reading a reference to some article or book. He gives the name or names identifying the article or book and the page or line number in which he is interested, and the desired text is quickly displayed for him so that his reading can continue without appreciable interruption. The technique of program segmentation appears to have many other advantages beyond those suggested here, and these are currently being explored in a number of research laboratories. Segmentation makes it possible for several central processors to combine in working on a program involving much computation and improves intercommunication within the system [*see lower illustration on page 138*]. Two new commercial computers, the General Electric Company's 645 and the IBM 360/67, include the special features needed for paging and for program segmentation.

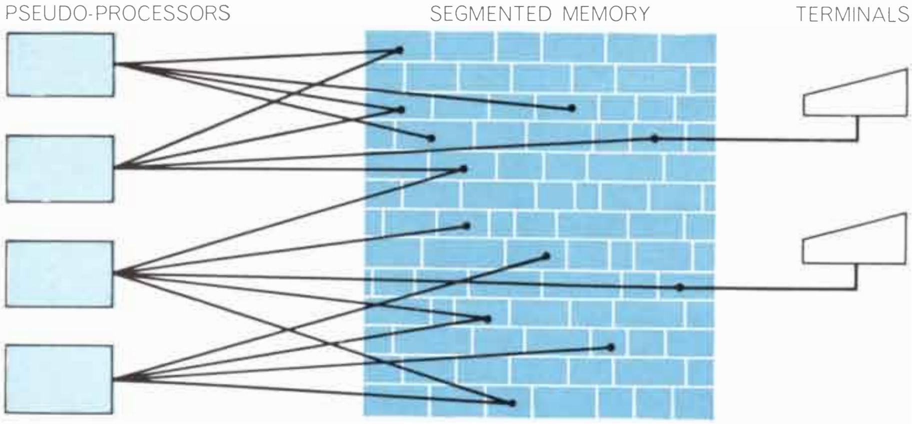
Finally, in this catalogue of improvements needed to develop time-sharing computers into general intellectual utilities we must mention a bottleneck for which a practicable solution is not yet in sight. The output devices still leave a great deal to be desired. The teletypewriter is a frustratingly slow means of communication—and it cannot draw a picture. The graphical display devices that are currently available are expensive and require elaborate communication facilities. Inasmuch as, from the standpoint of convenience and of economics, efficient communication between the time-sharing system and its users will become at least as important as the operation of the system itself, this problem presents a crucial challenge to designers.

Three years of experience with the Compatible Time-Sharing System at M.I.T. have been a revelation in many ways. In a sense the system and its users have developed like a growing organism. Most striking is the way the users have built on one another's work and become dependent on the machine. More than half of the commands now written into the system were developed by the users rather than by the professionals charged with programming and developing the system. The users have very generally chosen to link up with one another's private files and the public files. Whereas in conventional computer

installations one hardly ever makes use of a program developed by another user, because of the difficulty of exchanging programs and data, here the ease of exchange has encouraged investigators to design their programs with an eye to possible use by other people. They have acted essentially as if they were writing papers to be published in technical journals. Indeed, the analogy is not far-fetched: an editorial board representing the community of users acts as a referee to pass on all new commands that are to be introduced into the system and on all



PROPOSED DESIGN for a time-sharing computer would provide more dependability and flexibility than present systems. There would be several elements of each kind, so that no one unit would be critical. The main memories would be central physically as well as functionally. And the supervisor would assign each task to available units as required.



SEGMENTATION OF PROGRAMS adds to the flexibility of a time-sharing system. Each public and private file becomes an independent segment stored in the main memory, each with its own list of authorized users. There are no pseudo-memories, since each pseudo-processor can communicate with a number of segments, some shared with other processors.

information that is to be stored in the public files.

All in all, the mass memories of our machines are becoming more and more like a community library. The users are beginning to complain about the difficulty of finding out just what the library contains and locating the items that may be of interest to them. The facility actually goes beyond a library's usual services. It already has a rudimentary mechanism whereby one person can communicate with another through a program in real time, that is, while both are using the same program at the same time. There have been cases in which a member of the faculty, sitting at a teletypewriter at home, has worked with a student stationed at a terminal on the campus. It is easy now to envision the use of the system for education or for real-time collaboration between the members of a research team. And it does not take a long stretch of the imagination to envision an entire business organization making and executing all its major decisions with the aid of a time-shared computing system. In such a system the mass memory at all times would contain an up-to-date description of the state of the business.

Looking into the future, we can foresee that computer utilities are likely to play an increasingly large part in human affairs. Communities will design systems to perform various functions—intellectual, economic and social—and the systems in turn undoubtedly will have profound effects in shaping the patterns of human life. The coupling between such a utility and the community it serves is so strong that the community is actually a part of the system itself. Together the computer systems and the human users will create new services, new institutions, a new environment and new problems. It is already apparent that, because such a system binds the members of a community more closely together, many of the problems will be ethical ones. The current problem of wiretapping suggests the seriousness with which one must consider the security of a system that may hold in its mass memory detailed information on individuals and organizations. How will access to the utility be controlled? Who will regulate its use? To what ends will the system be devoted, and what safeguards can be devised to prevent its misuse? It is easy to see that the progress of this new technique will raise many social questions as well as technical ones.